docker COMMAND CHEAT SHEET

RUNNING CONTAINERS

`docker run <image-name>` Start a container from an image

`docker run --name <name>` Give a container a friendly name

`docker run -p <outside-port>:<inside-port>`
Publish a container's port on the host

`docker run --rm` Automatically remove a container when it stops

`docker run -d` Run a container in *detached* mode

`docker run -p <image> <command>`
Run a command within the container when the container starts

`docker run --entrypoint <entrypoint>`
Start a container with a custom entry point

`docker run -e <ENV-VARIABLE>=<value>`
Pass an environment variable to the container at runtime

`docker run --labels <label>`
Add a label to a container at runtime

INTERACTIVE COMMANDS

`docker run -it` Start in interactive mode, with a *pseudo-tty*

`docker exec <container> <command>`
Send a command to a container

`docker exec -it` Connect to the container interactively

CONTAINER MANAGEMENT

`docker create <image>`	Create a container, but don't start it
`docker start <container>`	Start a container that's in the *created* state
`docker start <container>`	Start a container that's in the *created* state
`docker stop <id>`	Stop a running container by it's ID
`docker stop <name>`	Stop a running container by its name
`docker pause <container>`	Pause a *running* container
`docker unpause <container>`	Unpause, or resume, a *paused* container
`docker ps`	Show all *running* containers
`docker ps -a`	Show all containers, in any state
`docker rm <id>`	Remove a stopped container, by its ID
`docker rm <name>`	Remove a stopped container, by its name
`docker inspect <container>`	Get information about a running container

NETWORK

```
docker network create <name>
```
Create a bridge network

```
docker network create -d macvlan
    -o parent=<host-NIC> <name>
```
Create a MACVLAN network

```
docker network create -d ipvlan -o ipvlan_mode=l2
    -o parent=<host-NIC> <name>
```
Create an IPVLAN network in L2 mode

```
docker network create -d ipvlan -o ipvlan_mode=l2
    -o parent=<host-NIC>.<vlan> <name
```
Create an IPVLAN network on a specific VLAN

```
docker network create -d ipvlan -o ipvlan_mode=l3
    -o parent=<host-NIC> <name>
```
Create an IPVLAN network in L3 mode

```
docker network create ... --subnet=<CIDR>
```
Assign a subnet to a network

```
docker network create ... --gateway=<ip>
```
Assign a default gateway to a network

```
docker network ls
```
Show a list of all networks on the host

```
docker network inspect <name>
```
Get more information about a network

```
docker run --network <name>
```
Assign a container to a network at runtime

```
docker run --ip <ip>
```
Assign an IP address to a container at runtime

docker COMMAND CHEAT SHEET

IMAGE MANAGEMENT

`docker image list`	Show a list of images available locally
`docker image inspect <image>`	Show additional information about an image
`docker pull <repository>:<tag>`	Pull an image from Docker Hub
`docker tag <image> <username>/<image>:<tag>`	Add tags to an existing image
`docker login -u <username>`	Login to an image registry, like Docker Hub
`docker push`	Push an image to a registry
`docker logout`	Logout from a registry
`docker system df`	Show disk space used by images

BUILD

`docker build . -t <image>:<tag>`	Build an image from a dockerfile in the working directory
`docker build -f <dockerfile>`	Build an image using a specific dockerfile
`docker history <image>`	See the layers of an image

STORAGE

`docker volume create <name>` Create a named volume

`docker volume list` Show a list of volumes on the host

`docker volume inspect <volume>`
Get more information about a volume

`docker volume rm <id>` Remove a volume, by its ID

`docker volume rm <name>` Remove a volume, by its name

`docker run --volume <volume>:<mount-point>`
Mount a named volume at runtime

`docker run --mount type=volume,src=<name>,dst=<mount>`
Alternate method to mounting a volume

`docker run --volume <mount-point>`
Add an unnamed volume at container runtime

`docker run --volume <host-directory>:<mount-point>`
Create a bind mount at runtime

docker COMMAND CHEAT SHEET

COMPOSE

`docker compose up`	Start an application from a compose file
`docker compose up -d`	Start an application in detatched mode
`docker compose up --build`	Rebuild any container with a build section
`docker compose up --force-recreate`	Restart the containers in the application
`docker compose ls`	See compose apps that are running
`docker compose down`	docker compose down
`docker compose version`	Get the version of docker compose
`docker compose logs`	Show logs from a compose application
`docker compose logs [-f]`	Show compose logs in real time

LOGGING

`docker logs <container>`	Get logs from a container
`docker logs --follow <container>`	Get logs as they are generated in real-time
`docker logs --follow <container>`	Show logs within a timeframe
`docker logs --timestamps <container>`	Add timestamps to logs

SECURITY

`docker run ... --read-only` Set the filesystem to read only

`docker run ... --cap-drop <capability>`
Drop a Linux capability

`docker run ... --cap-drop ALL`
Drop all capabilities

`docker run ... --cap-add <capability>`
Add a Linux capability

`docker run ... --privileged` Runs a container with virtually no isolation

`docker run -u $(id -u):$(id -g)`
Run a container with a specific UID

`docker run --pid <mode>` Set the process ID namespace for the container

RESOURCES AND LIMITS

`docker stats <container>` Check container resource usage

`docker run --memory=<value>m` Set the memory limit on a container

`docker run --memory=<value> --memory-swap=<value>`
Set physical and swap memory limits

`docker run --device-read-bps <disk>:<limit>`
Limit disk reads by bytes per second

`docker run --device-write-bps <disk>:<limit>`
Limit disk writes by bytes per second

`docker run --device-read-iops <disk>:<limit>`
Limit disk read by IOPS

`docker run --device-write-iops <disk>:<limit>`
Limit disk write by IOPS

`docker run --cpus="<value>"` Hard limit to CPU cores

`docker run --cpu-shares=<value>`
Soft CPU limit based on shares

`docker run --cpu-period=<value> --cpu-quota=<value>`
Hard CPU limit by time slices

`docker run --pids-limit <value>`
Set the limit of processes that can start

`docker run --ulimit nofile=<soft>:<hard>`
Soft and hard limits on the number of file descriptors

`docker run --ulimit nproc=<soft>:<hard>`
Soft and hard limits on the number of processes

docker Dockerfile Reference

BASIC DIRECTIVES

`FROM`	Select the base image to start from
`RUN`	Run a command during the build Commonly used to install packages, add users, create directories, set file permissions
`COPY`	Copy files from the source system into the image

ENTRYPOINTS

`CMD`	Acommand to run when the container first starts. Indended to be a default command, which is easily overridden. May be written in shell form or exec form.
`ENTRYPOINT`	Very similar to CMD Not intended to be as easily overridden Can be combined with CMD

docker Dockerfile Reference

METADATA

`EXPOSE`	Provides information about which network ports are used in the container
`LABEL`	An annotation that describes an image A key-value pair

VARIABLES

`ENV`	Define environment variables within the container Key-value pair
`ARG`	A variable used within the docker file Used for the build process only Does not become a variable in the final image

ENVIRONMENT

`WORKDIR`	Sets the working directory within the container The default location for commands to run from Like a home directory when you use regular linux
`USER`	Set the active user for the container Commands before this statement are run as root Build commands run after this statement are run as this user

docker Compose Reference

MAIN STRUCTURE

`name:`	Optional friendly name for the project
`services:`	A group of services (containers). The name of each service is user-defined.
`container_name:`	Optional friendly name for a container within a service
`image:`	The image used to start the container. Optionally includes a project name, repository, and tag.
`restart:`	The restart policy. Options are no, always, on-failure, unless-stopped
`command:`	Override the container's default command. Used to customise the container's start up behaviour
`extra_hosts:`	Provides a static hostname to IP mapping

```
name: nginx-webserver
services:
  nginx:
    container_name: nginx-web
    image: nginx:latest
    command: [nginx, "-g", "daemon off;"]
    extra_hosts:
      - "cloudflare-dns:1.1.1.1"
    restart: unless-stopped
```

docker Compose Reference

NETWORKING

`networks:`	The top level area where a network is defined
`driver:`	Set the driver type for a network
`external:`	When true, this indicates that the network has already been created, separate to this compose file.

`networks:`	Under a service, this is the area to add a list of networks.
`ipv4_address:`	Assign a static IPv4 address
`ports:`	Expose, or 'publish' container ports on the host

NETWORKING - EXAMPLE

```
services:
  web-app:
    networks:
      - internal
    ports:
      - 80:80

networks:
  internal:
    driver: bridge
```

docker Compose Reference

STORAGE

`volumes:`	The top level area where named volumes are defined.
`<name>:`	A named volume. No additional parameters are needed.

`volumes:`	Under a service, this is the area to add a list of volumes.
`read-only:`	When true, the root filesystem of the container is read-only.
`tmpfs:`	Create a virtual filesystem in memory.

STORAGE - EXAMPLE

```
services:
  my-service:
    volumes:
      - data:/data            # Named volume
      - /cache                 # Anon volume
      - /local:/config:ro   # Bind mount, read only
    read_only: true        # Read only FS
    tmpfs:
      - /tmp

volumes:
  data:
```

ENVIRONMENT VARIABLES

`environment:` Area within the service to place environment variables.

`<ENV>=` Create an environment variable and assign a value

`${VAR}` Reference an environment variable

`${VAR:-default}` Set a default value, or use an environment variable if it exists

ENVIRONMENT VARIABLES - EXAMPLE

```
services:
  special-app:
    environment:
      URL=networkdirection.net
      TZ=Australia/Sydney
      TAG=v1

    # Default to 'latest' if 'TAG' not set
    image: myImage:${TAG:-latest}
```

HEALTH CHECKS

`healthcheck:`	Area within the service to configure health checks.
`test:`	The command to use to determine health.
`interval:`	Time between checks. Default is 30s.
`timeout:`	Max time to wait before marking the test as failed. Default is 30s.
`retries:`	Number of consecutive failures before marking the container as unhealthy. Defaults to 3.
`start_period:`	Grace period between container start and health check start.
`depends_on:`	Configure a service to rely on another service. May optionally set health conditions.

HEALTH CHECK - EXAMPLE

```
services:
  healthy-container:
    healthcheck:
      test: ["ping", "127.0.0.1"]
      interval: 10s
      timeout: 4s
      retries: 5
      start_period: 10s
```

SECURITY

`cap_drop:`	Drop some or all Linux capabilities.
`cap_add:`	Add Linux capabilities to a container.
`sysctls:`	Linux system controls. Configure kernel modules.

SECURITY - EXAMPLE

```
services:
  secure-container:
    cap_drop:
      - ALL
    cap_add:
      - NET_ADMIN
    sysctls:
      net.ipv4.conf.all.arp_ignore: 1
```

docker Compose Reference

BUILD

`build:`	The area in the service that contains build information.
`context:`	The location of the files used in the build.
`dockerfile:`	The location of the dockerfile.

SECURITY - EXAMPLE

```
services:
  custom-container:
    build:
      context: .
      dockerfile: Dockerfile
```

RESOURCE LIMITS

`deploy:`	The new location for some limits.
`memory:`	The newer way to limit memory usage; Under 'deploy'.
`reservations:`	Reserve memory; Under 'deploy'.
`mem_limit:`	The older way to set a memory limit; Under 'services'.

`ulimits:`	Limit open files and processes.
`nofile:`	Set the max number of open file descriptors.
`nproc:`	Set the max number of open processes.

`blkio_config:`	Set block IO device limits (hard drives)
`device_read_iops`	Limit read IOPS
`device_write_iops:`	Limit write IOPS
`device_read_bps`	Limit read BPS
`device_write_bps:`	Limit write BPS

`mem_swap_limit:`	Set a limit on swap memory

RESOURCE LIMITS - EXAMPLE

```
services:
  limited-container:
    deploy:
      resources:
        limits:
          memory: 512M
        reservations:
          memory: 256M
    ulimits:
      nofile:
        soft: 1024
        hard: 2048
      nproc: 128
    blkio_config:
      device_read_iops
        /dev/sda:100
      device_write_iops
        /dev/sda:100
      device_read_bps:
        /dev/sda:5mb
      device_write_bps:
        /dev/sda:5mb
    memswap_limit: 512M
```